National College of Ireland

# Configuration Manual

MSc Research Project
Data Analytics


Arkoprovo Sarkar
Student ID: x19148038

School of Computing
National College of Ireland


Supervisor: Aaloka Anant

**National College of Ireland**
**Project Submission Sheet**
**School of Computing**

| | |
|---|---|
| **Student Name:** | Arkoprovo Sarkar |
| **Student ID:** | x19148038 |
| **Programme:** | Data Analytics |
| **Year:** | 2022-2023 |
| **Module:** | MSc Research Project |
| **Supervisor:** | Aaloka Anant |
| **Submission Due Date:** | 15/12/2022 |
| **Project Title:** | Configuration Manual |
| **Word Count:** | 490 |
| **Page Count:** | 8 |

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

**ALL** internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

| | |
|---|---|
| **Signature:** | |
| **Date:** | 15th December 2022 |

## PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

| | |
|---|---|
| Attach a completed copy of this sheet to each project (including multiple copies). | ☐ |
| **Attach a Moodle submission receipt of the online project submission**, to each project (including multiple copies). | ☐ |
| **You must ensure that you retain a HARD COPY of the project**, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer. | ☐ |

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

| **Office Use Only** | |
|---|---|
| Signature: | |
| Date: | |
| Penalty Applied (if applicable): | |

# Configuration Manual

Arkoprovo Sarkar
x19148038

## 1 Introduction

The actions that must be taken in order to successfully execute the scripts that were used during the research are outlined in this document. You will be able to successfully execute the code with the assistance of these instructions. In addition to that, information on the hardware setup of the machine that was used to execute the code has been included in this document.

## 2 System Specifications

### 2.1 Hardware Requirements

This research was carried out on a Dell Vostro 14 laptop. The laptop's hardware specifications are as follows:

- Processor: Intel Core i5 10th Generation
- Operating System: Windows 10
- RAM: 16 GB
- Storage: 256 GB

### 2.2 Software Requirements

During the course of this research, the following tools were used:

- Python version 3.8
- Microsoft Excel
- Power BI

## 3 Setting Up the Environment

The whole study was carried out using Python script in Jupyter Notebook on Anaconda Navigator.

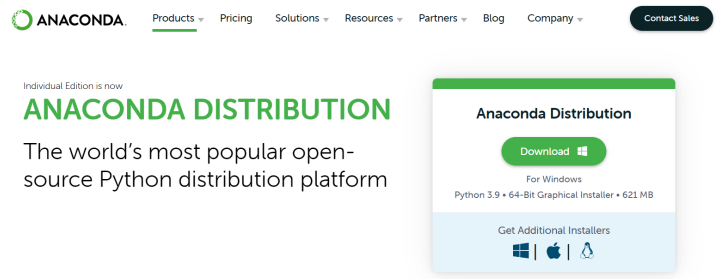


Figure 1: Anaconda Distribution

Anaconda Navigator was opened once it was installed successfully. Then Jupyter Notebook was launched using the Google Chrome browser as its basic platform.

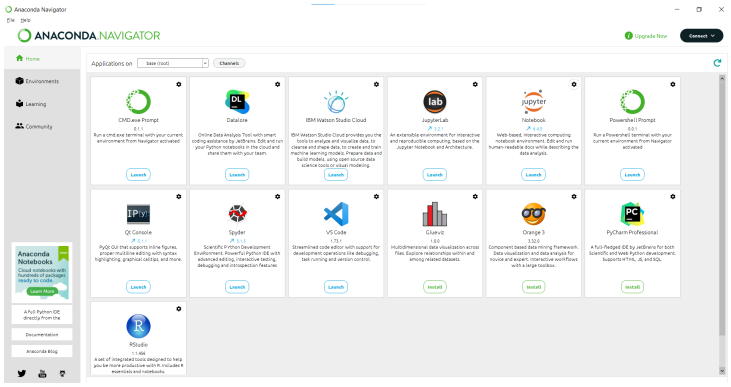


Figure 2: Anaconda Navigator

# 4 Selection of Data

The dataset was gathered from Kaggle.

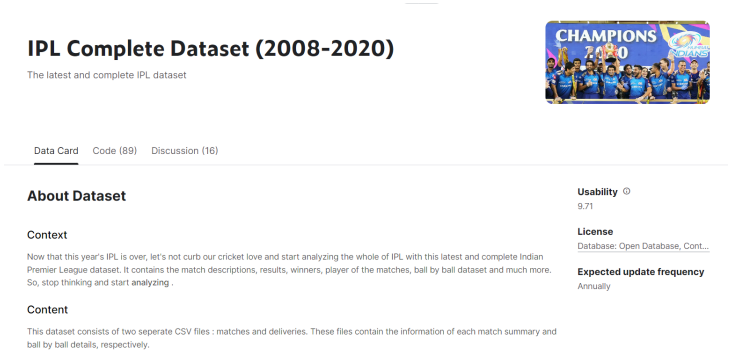


Figure 3: IPL Dataset

Age of the players was not included in the dataset. Therefore, it had to be inserted manually. The final dataset with players age has been shared together with the code. That must be used as source data.

# 5 Implementation

## 5.1 Code Blocks

### Import Required Libraries

The libraries necessary for executing Batsman_Dataset_VFinal script are mentioned below:

```
#importing libraries
import os
import warnings
import pandas as pd
import numpy as np
from sklearn.preprocessing import LabelEncoder
from matplotlib import pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn import metrics
from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
from sklearn.metrics import f1_score
```

Figure 4: List of Required Libraries

### Reading Data

In order to read the data, Pandas data frame was used, as shown below.

```
#reading ball by ball dataset
ipl_ballbyball= pd.read_csv("C:/Users/arkos/OneDrive/Documents/Masters/Research Project/IPL_Dataset_Final/Ball-by-Ball.csv")
ipl_ballbyball.head()
```

```
#reading match details dataset
ipl_matches = pd.read_csv("C:/Users/arkos/OneDrive/Documents/Masters/Research Project/IPL_Dataset_Final/Match_Details.csv")
ipl_matches.head()
```

Figure 5: Data Reading

### Data Preprocessing

After reading the datasets, they were merged in one single dataframe and then the team names were updated.

```
#Creating a merged dataframe & dropping the columns which are not required columns
combined_ipl_data = pd.merge(ipl_matches, ipl_ballbyball, left_on='id', right_on='id',how='outer')
combined_ipl_data = combined_ipl_data.drop(['umpire1','umpire2'], axis = 1)
combined_ipl_data.head()
```

```
#delhi daredevils changed their team name to delhi capitals. Therefore, old team name has been updated with the new name.
combined_ipl_data = combined_ipl_data.replace(to_replace ="Delhi Daredevils", value = "Delhi Capitals")
#deccan chargers changed their team name to Sunrisers Hyderabad. Therefore, old team name has been updated with the new name.
combined_ipl_data = combined_ipl_data.replace(to_replace ="Deccan Chargers", value = "Sunrisers Hyderabad")
```

```
#Rising pune supergiants name has been printed in two ways. Merging them as both of them are same team.
combined_ipl_data = combined_ipl_data.replace(to_replace ="Rising Pune Supergiant", value = "Rising Pune Supergiants")
```

Figure 6: Merging Dataframes & Correcting Team Names

## Feature Engineering

Batting attributes were added one by one in the dataframe. Sample code has been highlighted in the following figure.

**Batsman Dataset Creation**

```
#runs scored
runs_scored=combined_ipl_data.groupby(['id','batsman'])['batsman_runs'].sum()
df_runs_scored = runs_scored.to_frame().reset_index()
df_runs_scored.rename(columns = {'batsman_runs' : 'runs_scored'}, inplace = True)
df_runs_scored.head(1)
```

```
#balls_faced
balls=combined_ipl_data[(combined_ipl_data['extra_runs'] == 0)]
balls_faced= balls.groupby(['id','batsman'])['batsman_runs'].count()
df_balls_faced = balls_faced.to_frame().reset_index()
df_balls_faced.rename(columns = {'batsman_runs' : 'balls_faced'}, inplace = True)
df_balls_faced.head(1)
```

```
#boundaries_scored
boundaries = combined_ipl_data[combined_ipl_data['batsman_runs'] == 4]
boundaries_scored= boundaries.groupby(['id','batsman'])['batsman_runs'].count()
df_boundaries_scored=boundaries_scored.to_frame().reset_index()
df_boundaries_scored.rename(columns = {'batsman_runs' : 'boundaries_scored'}, inplace = True)
df_boundaries_scored.head(1)
```

```
#sixes hit
sixes = combined_ipl_data[combined_ipl_data['batsman_runs'] == 6]
sixes_scored= sixes.groupby(['id','batsman'])['batsman_runs'].count()
df_sixes_scored=sixes_scored.to_frame().reset_index()
df_sixes_scored.rename(columns = {'batsman_runs' : 'sixes_hit'}, inplace = True)
df_sixes_scored.head()
```

```
#merging
merged_df = df_runs_scored.merge(df_balls_faced, how = 'inner', on = ['id', 'batsman'])
#calculating strike rate
merged_df['strike_rate'] = round(((merged_df['runs_scored'] / merged_df['balls_faced']) * 100),2)
merged_df.head(1)
```

```
#adding no of innings for each game
merged_df['no_of_innings'] = 1

merged_df=merged_df.merge(df_boundaries_scored, how = 'outer', on = ['id', 'batsman'])
merged_df=merged_df.merge(df_sixes_scored, how = 'outer', on = ['id', 'batsman'])
merged_df["boundaries_scored"].fillna(0, inplace = True)
```

Figure 7: Batting Attributes

```
#caculating consistency by using the equation derived by Passi and Pandey in thier study
#also calculating overall average & overall strike rate
def attribute(df,col_name):
    df['avg_runs']=0.0
    index_ba=df.columns.get_loc("avg_runs")
    index_in=df.columns.get_loc("no_of_innings")
    index_inruns=df.columns.get_loc("runs_scored")
    for row in range(len(df)):
        inumber=df.iat[row,index_in]
        inruns=df.iat[row,index_inruns]
        df.iat[row,index_ba]=inruns/inumber

    df['srate']=0.0
    index_ba=df.columns.get_loc("srate")
    index_in=df.columns.get_loc("balls_faced")
    index_inruns=df.columns.get_loc("runs_scored")
    for row in range(len(df)):
        inumber=df.iat[row,index_in]
        inruns=df.iat[row,index_inruns]
        df.iat[row,index_ba]=(inruns/inumber)*100

    index_new=df.columns.get_loc(col_name)
    index_sr=df.columns.get_loc("srate")
    index_av=df.columns.get_loc("avg_runs")
    index_in=df.columns.get_loc("no_of_innings")
    index_100=df.columns.get_loc("100s")
    index_50=df.columns.get_loc('50s')
    index_0=df.columns.get_loc('ducks')

    for row in range(len(df)):
        f=0.4262*(df.iat[row,index_av])
        f=f+0.2566*(df.iat[row,index_in])
        f+=0.1510*(df.iat[row,index_sr])
        f+=0.0787*(df.iat[row,index_100])
        f+=0.0556*(df.iat[row,index_50])
        f=f-(0.0328*(df.iat[row,index_0]))
        df.iat[row,index_new]=f

    return(df)



g=batter_data_df.groupby('batsman')
df=g.sum()
df['consistency']=0.0
df=attribute(df,'consistency')
df['overall_average']=df['avg_runs']
df['overall_strike_rate']=df['srate']
drop=['age','id','runs_scored','boundaries_scored','sixes_hit', 'strike_rate','innings_no','no_of_innings' ,'balls_faced', 'year'
df.drop(drop,axis=1,inplace=True)
batter_data_df=pd.merge(batter_data_df,df,on='batsman', how='inner')
```

Figure 8: Derived Attributes

The players were shortlisted depending on their ages.

```
#Removing players of age greater than 30 (as of 2020)
batter_data_df_bin_lessthan30 = batter_data_df_bin_lessthan30[batter_data_df_bin_lessthan30.age <= 30]
```

Figure 9: Shortlisting Players

## Data Binning

Here each batting attribute was given a value between one and five, with one being the least significant and five representing the most essential.

**BINNING KEY ATTRIBUTES**

```
# binning Consistency
tempData = []
dummy_con = []

for i in range (0,len(batter_data_df_bin.index)):
    tempData.append('temp')
batter_data_df_bin['consistency_bin']=tempData

for i in range (0,len(batter_data_df_bin.index)):
        if batter_data_df_bin['consistency'][i] <= 20:
            batter_data_df_bin['consistency_bin'][i] = 1
        elif batter_data_df_bin['consistency'][i]>20 and batter_data_df_bin['consistency'][i] <=40:
            batter_data_df_bin['consistency_bin'][i] = 2
        elif batter_data_df_bin['consistency'][i] >40 and batter_data_df_bin['consistency'][i] <=50:
            batter_data_df_bin['consistency_bin'][i] = 3
        elif batter_data_df_bin['consistency'][i] >50 and batter_data_df_bin['consistency'][i] <=60:
            batter_data_df_bin['consistency_bin'][i] = 4
        elif batter_data_df_bin['consistency'][i] >60:
            batter_data_df_bin['consistency_bin'][i] = 5
```

```
#binning form
tempData = []
dummy_form = []

for i in range (0,len(batter_data_df_bin.index)):
    tempData.append('temp')
batter_data_df_bin['form_bin']=tempData

for i in range (0,len(batter_data_df_bin.index)):
        if batter_data_df_bin['form'][i] <= 20:
            batter_data_df_bin['form_bin'][i] = 1
        elif batter_data_df_bin['form'][i]>20 and batter_data_df_bin['form'][i] <=40:
            batter_data_df_bin['form_bin'][i] = 2
        elif batter_data_df_bin['form'][i] >40 and batter_data_df_bin['form'][i] <=60:
            batter_data_df_bin['form_bin'][i] = 3
        elif batter_data_df_bin['form'][i] >60 and batter_data_df_bin['form'][i] <=80:
            batter_data_df_bin['form_bin'][i] = 4
        elif batter_data_df_bin['form'][i] >80:
            batter_data_df_bin['form_bin'][i] = 5
```

Figure 10: Data Binning

## Data Encoding

Data that was not numerical in nature was changed into numerical value so that it could be more readily fitted into a machine learning model.

```
#converting dtypes
batter_data_df_bin["venue_wise_strike_rate_bin"] = batter_data_df_bin["venue_wise_strike_rate_bin"].astype(float)
batter_data_df_bin["opposition_wise_strike_rate_bin"] = batter_data_df_bin["opposition_wise_strike_rate_bin"].astype(float)
batter_data_df_bin["yearly_strike_rate_bin"] = batter_data_df_bin["yearly_strike_rate_bin"].astype(float)
batter_data_df_bin["strike_rate_bin"] = batter_data_df_bin["strike_rate_bin"].astype(float)
batter_data_df_bin["overall_strike_rate_bin"] = batter_data_df_bin["overall_strike_rate_bin"].astype(float)
batter_data_df_bin["venue_wise_average_bin"] = batter_data_df_bin["venue_wise_average_bin"].astype(float)
batter_data_df_bin["opposition_wise_average_bin"] = batter_data_df_bin["opposition_wise_average_bin"].astype(float)
batter_data_df_bin["yearly_average_bin"] = batter_data_df_bin["yearly_average_bin"].astype(float)
batter_data_df_bin["overall_average_bin"] = batter_data_df_bin["overall_average_bin"].astype(float)
batter_data_df_bin["opponent_bin"] = batter_data_df_bin["opponent_bin"].astype(float)
batter_data_df_bin["form_bin"] = batter_data_df_bin["form_bin"].astype(float)
batter_data_df_bin["consistency_bin"] = batter_data_df_bin["consistency_bin"].astype(float)
batter_data_df_bin["runs_bin"] = batter_data_df_bin["runs_bin"].astype(float)
batter_data_df_bin["venue_bin"] = batter_data_df_bin["venue_bin"].astype(float)
```

```
#Label Encoding

def encode(df,col):
    le = LabelEncoder()
    batter_data_df_bin[col] = le.fit_transform(df[col])

encode(batter_data_df_bin,'batsman')
encode(batter_data_df_bin,'ground')
encode(batter_data_df_bin,'opponent')
encode(batter_data_df_bin,'date')
```

Figure 11: Data Encoding

## Data Correlation

The pairwise correlation of each column was found with the help of the corr() function.

```
#Checking co-relation between columns
corr = batter_data_df_bin_lessthan30.corr()
fig, ax = plt.subplots(figsize=(30, 18))
colormap = sns.diverging_palette(220, 10, as_cmap=True)
dropSelf = np.zeros_like(corr)
dropSelf[np.triu_indices_from(dropSelf)] = True
colormap = sns.diverging_palette(220, 10, as_cmap=True)
sns.heatmap(corr, cmap=colormap, linewidths=.5, annot=True, fmt=".2f", mask=dropSelf)
plt.title('Runs - Features Correlations')
plt.show()
```

Figure 12: Correlation Between Columns

## Data Split and Machine Learning Model Evaluation

Finally the experiments were carried out using a range of training and test set sizes in order to access the models.

**70:30 Split**

```
X_train2, X_test2, Y_train2, Y_test2 = train_test_split(runs_train, runs_target, test_size = 0.30, random_state = 0)
```

```
sc = StandardScaler()
X_train2 = sc.fit_transform(X_train2)
X_test2 = sc.transform(X_test2)
```

```
#Random Forest
model_rf.fit(X_train2, Y_train2)
Y_pred2=model_rf.predict(X_test2)

print('Training Accuracy:', model_rf.score(X_train2, Y_train2))
print(F'Accuracy:',accuracy_score(Y_test2, Y_pred2))
print(F'Precision:', precision_score(Y_test2, Y_pred2,average='weighted'))
print(F'Recall:', recall_score(Y_test2, Y_pred2,average='weighted'))
print(F'F1 Score:', f1_score(Y_test2, Y_pred2,average='weighted'))
```

```
#Naive Bayes
nb_model.fit(X_train2,Y_train2)
y_pred2=nb_model.predict(X_test2)

print('Training Accuracy:', nb_model.score(X_train2, Y_train2))
accuracy=accuracy_score(Y_test2,y_pred2)
precision=precision_score(Y_test2,y_pred2,average='weighted')
recall=recall_score(Y_test2,y_pred2,average='weighted')
f1=f1_score(Y_test2,y_pred2,average='weighted')

print('Accuracy - {}'.format(accuracy))
print('Precision - {}'.format(precision))
print('Recall - {}'.format(recall))
print('F1 - {}'.format(f1))
```

Figure 13: 70:30 Data Split & Model Evaluation

**80:20 split**

```
X_train3, X_test3, Y_train3, Y_test3 = train_test_split(runs_train, runs_target, test_size = 0.20, random_state = 0)
```

```
sc = StandardScaler()
X_train3 = sc.fit_transform(X_train3)
X_test3 = sc.transform(X_test3)
```

```
#Random Forest
print(model_rf.fit(X_train3, Y_train3))
Y_pred3=model_rf.predict(X_test3)

print('Training Accuracy:', model_rf.score(X_train3, Y_train3))
print(F'Accuracy:',accuracy_score(Y_test3, Y_pred3))
print(F'Precision:', precision_score(Y_test3, Y_pred3,average='weighted'))
print(F'Recall:', recall_score(Y_test3, Y_pred3,average='weighted'))
print(F'F1 Score:', f1_score(Y_test3, Y_pred3,average='weighted'))
```

```
#Naive Bayes
nb_model.fit(X_train3,Y_train3)
y_pred3=nb_model.predict(X_test3)

print('Training Accuracy:', nb_model.score(X_train3, Y_train3))
accuracy=accuracy_score(Y_test3,y_pred3)
precision=precision_score(Y_test3,y_pred3,average='weighted')
recall=recall_score(Y_test3,y_pred3,average='weighted')
f1=f1_score(Y_test3,y_pred3,average='weighted')

print('Accuracy - {}'.format(accuracy))
print('Precision - {}'.format(precision))
print('Recall - {}'.format(recall))
print('F1 - {}'.format(f1))
```

Figure 14: 80:20 Data Split & Model Evaluation

A similar procedure was followed while running the Bowlers_Dataset_VFinal script.